A search engine model: Web Scraping, NaturalLanguage Processing, and Pointwise Mutual Information

1st Josue´ Genaro Almaraz Rivera
School of Engineering and Sciences
Tecnológico de Monterrey
Monterrey, México
a00821189@itesm.mx

2nd Francisco Javier Cantú Ortiz
School of Engineering and Sciences
Tecnológico de Monterrey
Monterrey, México
fcantu@tec.mx

3rd Héctor Gibrán Ceballos Cancino
School of Engineering and Sciences
Tecnológico de Monterrey
Monterrey, México
ceballos@tec.mx

***Abstract*—This document proposes a search engine model which applies Web Scraping, Natural Language Processing (NLP), and Pointwise Mutual Information (PMI), for extracting and analyzing websites content. It follows the main purpose of settle down the fundamentals for search engine designs, to allow anyone who is interested in computing and linguistics to dive into this field. Maybe with the proposed solution, more people will be motivated to improve the knowledge access of many different disciplines.**

***Index Terms*—Web Crawling, Web Scraping, Search Engine, Natural Language Processing, Pointwise Mutual Information**

I. Introduction

There is many information right now on the web, and the need for answering new questions increases every day. There are search engines focused on very specific topics, like those of academia and science (e.g. ACM Digital Library and EBSCO), and those of a wider field which surely are the first that come to our mind (e.g. Google and Bing). Even, in a website or app (e.g. blogs and marketplaces), we can find search engines crawling articles and products.

The search engines design has experienced an interesting evolution over the years, and to identify the fundamentals, or the first steps to start in this area, seems quite complicated with the vast information that already exists. In this document, an approach from the basics scaling up to more complex concepts is presented, inviting more people to explore computing and linguistics.

The research questions that are planned to answer are:

1) What is the state of the art in search engines design?
2) Is Web Scraping and Natural Language Processing (NLP) a good combination for information extraction and processing?
3) What can Pointwise Mutual Information (PMI) measure tell us about a website content?
4) What are the next challenges for information retrieval and questions answering over the Internet?

II. Methodology and Data

As any field in Computer Science, applied mathematics are needed. The main topics that will be covered in this article are Graph Theory and Regular Expressions. Automata Theory is also fundamental for understanding the search engines design, but in this document just some remarks are mentioned. As said by Hopcroft et al. [1]: Finite State Automata constitute an useful model for a lot of hardware and software. One of the main applications of these concepts, are software for exploring large amounts of text, as websites collections, or to determine the number of occurrences of a word, phrase, or any other patterns.

A common nowadays problem is to determine *those places where a set of words appear*. A **search engine** comes into scene to solve this problem.

When someone talks about what strategies this kind of software implements, one that can be mentioned is called *Inverted Index*. Heo et al. [2] explain this concept in their recent article called *IIU: Specialized Architecture for Inverted Index Search*: Inverted index serves as a fundamental data structure for efficient search across various applications such as full- text search engine, document analytics and other information retrieval systems.

But there is a problem with this technique: the requirements of storage and query load for these structures have been rapidly growing. Work has been done mainly focused on CPUs and GPUs to exploit query parallelism with top compression schemes, but it still been challenging to fit the index in memory (for millions of words). It is pretty tough to create and store a list of all the places in the web where a word exists. Even when we have large amount of memory, we need to keep available the more frequent lists, to allow our users to search simultaneously.

Jun Heo and his team, present a solution they called **IIU**, which is a novel inverted index processing unit, both indexing scheme and hardware accelerator, so that it can process highly compressed inverted index at a high throughput.

Having in mind all these technical problems, a question that really comes to mind is: what is the technology behind the famous search engines that we interact everyday? In the next paragraphs, this question is answered by the Google case study.

A. Google case study: the technology behind

Google has done good efforts in making public their knowledge about how to extract information from the World Wide Web; they have a public website where they explain how their search works [3], and even a guideline they use to evaluate their search engine algorithms quality through ratings [4].

Google explains they use software known as **web crawlers** that look at web pages and follow links on those pages (much like a person would do), bringing data back to Google's servers. This process begins with a list of hundreds of bil-lions of web addresses (known as the **Search index**), which according to them, is over 100,000,000 gigabytes in size.

While doing this search, **Google ranking systems** sort through these web pages to find the most relevant and useful results, in literally, a fraction of a second. This ranking considers factors such as:

- The words of the query.
- Relevance of the page.
- Usability of the page.
- User location and settings.
- Expertise of sources.

Not the same weight is applied to each of these factors. It varies depending on what we are looking for (i.e. *the natureof the query*). For instance, when we are searching current news topics, the freshness of the content matters the most.

But finding and organizing information by crawling and indexing is not enough. Presenting this information in many useful formats (e.g. maps, images, videos) is something pretty important to also care about, because it helps users to find what they're looking for quickly. This is what is known by Google as **useful responses**.

A mandatory question after all this, is: what sources of data Google uses? Well, they use an own **Knowledge Graph** that has millions of entries that describe people, places, and things (all of them nodes of the graph). Google uses also another sources, such as their vast collection of books (through **Google Books**), and the **World Bank**.

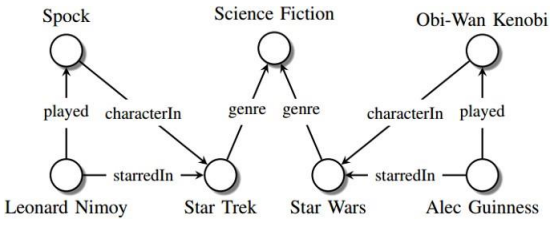

Fig. 1. An example of Knowledge Graph about science fiction, extracted from [5].

The use of the Knowledge Graph can be easily detected in the **knowledge panel** next to the web results. When we search the name of, for example, Bill Gates, we can see what people or even books are also related to him (i.e. the connections between entities in our graph).

What can be concluded here is that the success of Google relies upon its continuous search algorithms improvement

Fig. 2. Knowledge panel for Bill Gates.

to achieve high quality standards specified in their public guidelines. The efforts of the people behind and the applying of modern Artificial Intelligence and modern data extraction techniques, are key factors for this accomplishment.

B. So, what's the design of the here presented search engine proposal?

As has been stated, there are common key factors that increment the probability of success for this kind of software. The research here presented neither try to reinvent the wheel, nor to make the next "big thing" in the market. It aims to settle down the fundamentals for anyone who wants to start a research in these projects, know where to start and the minimum requirements to get something more than decent as a result.

The here proposed search engine design considers the next two features:

1) **Artificial Intelligence**: to achieve this is a complicated task, but a good approximation can be gotten by applying Natural Language Processing.
2) **Data Science**: there are millions of web pages all over the world. To design a web crawler, accompanied with web scraping, is a good start.

In the next lines the construction of this algorithm from scratch, and the results are shared, in order to discuss them and get some conclusions.

III. Results

A. Web scraping

The first of all, is to code a **spider**. Spider means, in this computer programming context, a class written to extract data from a website. The here proposed framework is **Scrapy**,which adds a pretty good feature on the table: *asynchronism*. This concept means that several users can make requests at thesame time to the spider, and it will not have trouble handling the job.

Two important questions come to mind:

1) What websites are going to be used?
2) What data is needed from the content of these websites?

Answering the first question, about what websites are going to be used, the proof of concept is done with Yahoo! Finance, and it pretends to scale up using the *web graph* shared by Common Crawl (a data set of over 85 million domains).

Common Crawl is a foundation based on California, whose goal is democratizing access to web information by producing and maintaining an open repository of web crawl data that is universally accessible and analyzable [6].

And now, answering the second question, the whole text of the website is going to be read and then filtered (e.g. removing the HTML tags) with the help of regular expressions and the Python Natural Language Toolkit (NLTK). As it is a Natural Language Processing task, there are linguistics concepts that need to be explained, and it will be done with key snippets of code samples of my own.

B. Natural Language Processing

Tokenization is a key concept in NLP, and it is defined as the process of extracting words from a text. In the code of the Figure 3 I apply a filter to our raw format (HTML), where I remove HTML tags and tokenize eliminating **stopwords** of the English language. Stopwords are words which when are alone, lack of sense, so I do not need them in the resulting tokens list. Also, I do not want words with length over than two (e.g. single numbers).

```python
import scrapy
import html2text
import re

import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords


class Content(scrapy.Spider):
    name = 'content'
    start_urls = [
        'https://finance.yahoo.com/'
    ]

    def parse(self, response):
        # get HTML from page
        html = response.text

        # remove HTML tags
        text = html2text.html2text(html)

        # tokenize the past resulting text, using regex
        tokens = re.findall('\w+', text)

        # filter by stopwords and word length
        stop_words = stopwords.words('english')
        tokens = [w for w in tokens if w not in stop_words and len(w) > 2]

        # analyze resulting tokens
        with open('test.txt', 'w') as f:
            f.write(str(tokens))
            f.write(f'\nLength: {str(len(tokens))}')
```

Fig. 3. Text normalization: tokenization of Yahoo! Finance home page.

All right, so now we have the function that is going to be applied to every website that needs to be scraped. It means that now we have clean data, and the next step is to treat the user query. How can this be achieved? Well, for explaining this, I will use **WordNet**, a lexical database of English, which is composed by **synsets** [7]. Synsets are sets of synonyms, andcan easily be explored by using NLTK. Let's say we want to find out the synsets for the word "business". In Figure 4 we can see the result:

```
business.n.01 :  a commercial or industrial enterprise and the people who constitute it
 *  business
 *  concern
 *  business_concern
 *  business_organization
 *  business_organisation
commercial_enterprise.n.02 :  the activity of providing goods and services involving financial and
commercial and industrial aspects
 *  commercial_enterprise
 *  business_enterprise
 *  business
occupation.n.01 :  the principal activity in your life that you do to earn money
 *  occupation
 *  business
 *  job
 *  line_of_work
 *  line
business.n.04 :  a rightful concern or responsibility
 *  business
```

Fig. 4. Some synsets in WordNet for the word *business*: *commercial enterprise* and *occupation* are some examples.

A graph is much better for illustrating the Figure 4. But for this, two key concepts need to be introduced:

- **Hyponyms**: these are words of more specific meaning with respect to the user query term (see Figure 5).
- **Hypernyms**: this is the opposite to an hyponym. Hypernyms are words with a broader meaning, and so, more specific words fall under (see Figure 6).

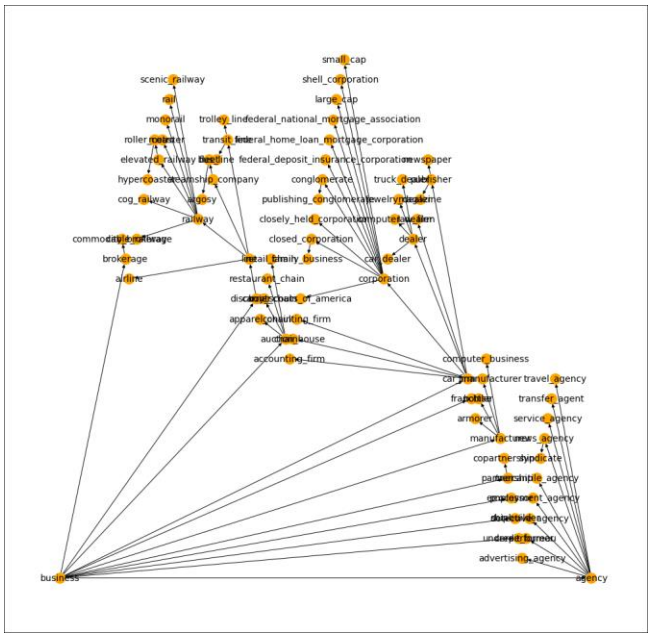

Fig. 5. Hyponyms in WordNet for the word *business*. Some examples are the words *agency*, *corporation*, and *accounting firms*.

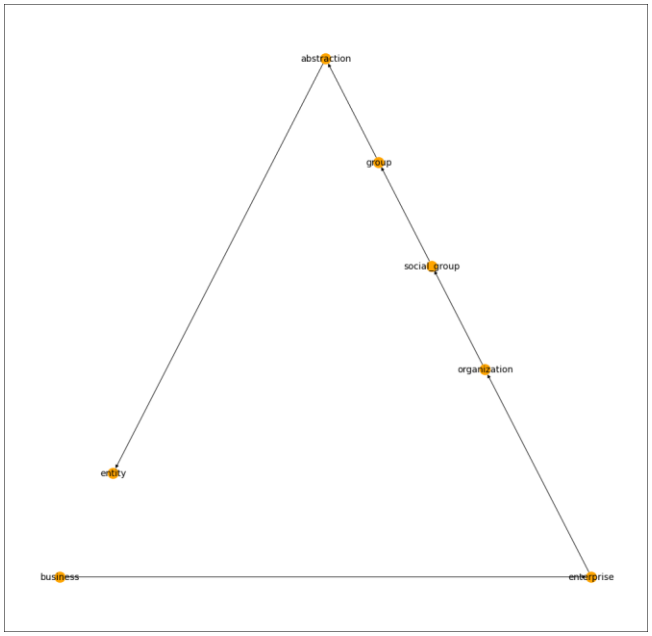

Fig. 6. Hypernyms in WordNet for the word *business*. The word *business* is related to *enterprises*, which at the same time are *organizations*, *social groups*, *groups*, *abstractions*, and at the end, *entities*.

With this information, it is more than clear that when an user types the word *business* in a search engine, it can really mean many things. The distance between nodes in these graphscan also be measured, with a concept called **path similarity**, whose code is in the Figure 7, and the output in Figure 8.Here when the result (distance) of two words is closer to 0, it means they are "more synonyms" than the words closer to 1.

```
groups = show_synsets('business')
groups_two = show_synsets('agency')
groups_three = show_synsets('corporation')

business = groups[0]
agency = groups_two[0]
corporation = groups_three[0]

print(business.path_similarity(agency))
print(business.path_similarity(corporation))
```

Fig. 7. Path similarity between *business* and *agency*, and *business* and *corporation*.

```
business and agency: 0.16666666666666666
business and corporation: 0.3333333333333333
```

Fig. 8. The word *business* is more closed to *agency*, rather than *corporation*.

C. Pointwise Mutual Information

Two fundamental concepts in linguistics, which help to answer the question *what is the main content of a text?*, are **n-grams** and **collocations**. *n-gram* is a sequence of *n* consecutive words, and *collocation* is a concept hard to define, but a simple explanation would be that it refers to the pair or groups of words with an abnormal frequency in a written or oral speech.

Once understanding that, NLTK can be used to list the bigrams (groups of two consecutive words) in the text of the website, filter these bigrams, get the frequency distribution, and finally, apply a measure called **Pointwise Mutual Information** (1), which helps to answer the question: *do words x and y co-occur more than if they were independent?* All thisin order to help determine if there exists a collocation or not.

$$PMI(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)} \quad (1)$$

Where $P(word_1)$ and $P(word_2)$ represent the probability of observing each of these words, and $P(word_1, word_2)$ is the probability of observing both $word_1$ and $word_2$ co-occurring.

The Figure 9 is a data frame of the bigrams, frequencies, and PMI of the text in the Yahoo! Finance home page. It is sorted in descending order by the PMI column.

	bigrams	word_0	word_1	bigram_frequency	word_0 frequency	word_1 frequency	PMI	log(bigram_frequency)
0	(Premium, Markets)	Premium	Markets	1	2	6	-3.584963	0.0
1	(empty, Watchlist)	empty	Watchlist	1	1	1	0.000000	0.0
2	(history, raising)	history	raising	1	1	1	0.000000	0.0
3	(selloff, probably)	selloff	probably	1	1	1	0.000000	0.0
4	(operating, model)	operating	model	1	1	1	0.000000	0.0
...	...	...	...	...	...	...	...	...
901	(wealth, management)	wealth	management	1	1	3	-1.584963	0.0
902	(think, NexTier)	think	NexTier	1	3	5	-3.906891	0.0
903	(consistent, management)	consistent	management	1	1	3	-1.584963	0.0
904	(positioned, rebound)	positioned	rebound	1	2	1	-1.000000	0.0
905	(Election, Watchlists)	Election	Watchlists	1	2	2	-2.000000	0.0

906 rows × 8 columns

Fig. 9. Data frame with the PMI results.

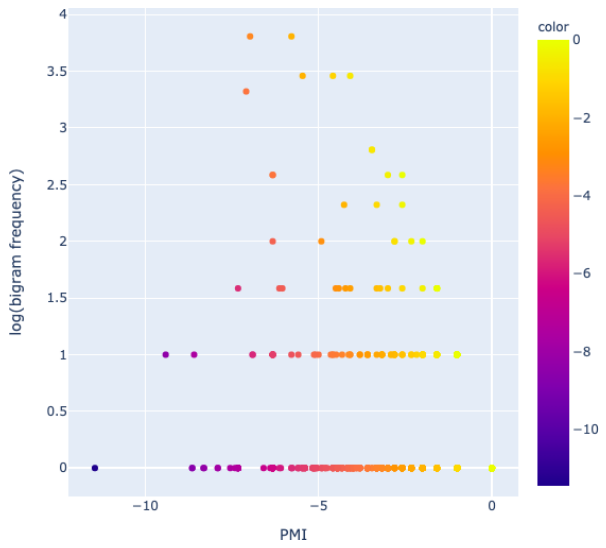

Fig. 10. PMI analyzed using a scatter graph.

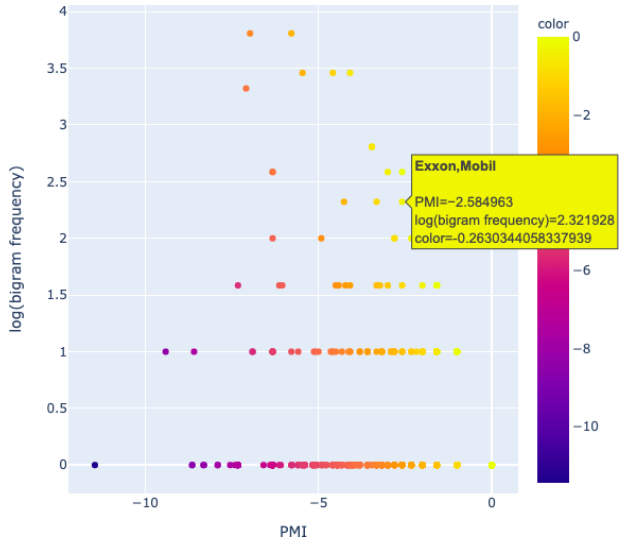

Fig. 11. The bigram (Exxon, Mobil) is one of the collocations in the analyzed text.

In Figure 10, the less negative PMI, and greater bigram frequency, the most probability of getting a collocation. As can be observed in Figure 11, ExxonMobil was a pretty commented company during that information analyzing day.

IV. Discussion

With the success of the results in section III, we are in a position to enumerate the steps of this search engine model proposal:

1) Iterate over the ranked domains in the web graph data set by the Common Crawl Foundation.
2) Respect the **robots.txt** file, which allows to not scrape content the website owners do not authorize. Web scraping is a beautiful technique, but it can generate some troubles if the ethical part is not taken into account.
3) Scrape the website to get the text of the home page.
4) Apply tokenization to extract all the words resulting of the text from the previous step.
5) Get the synonyms of the words present in the user query.
6) Get the bigrams and the collocations by applying NLTK and PMI to the tokenized text.
7) Compare the top resulting collocations with the user query and the synonyms from step 5, to determine if the website is going to be indexed and shown in the results.

By applying this approach, a good websites information extraction and analysis can be done for the purpose of building a search engine.

A. *The future: Insight engines*

When a Magic Cuadrant for search engines is looked for, Gartner Inc shows and explains a new current tendency: **Insight Engines**. Insight engines differ from search engines in terms of capabilities that enable richer indexes, more complex queries, elaborated relevancy methods, and multiple touch-points for the delivery of data (for machines) and information (for people) [8].

Insight engines reveal the following trends:

- **Acceptance of queries in natural language**, but the capability of generate natural language answers is still rising.
- **360 views**: providing users with all of the data around a person, business or topic.
- **Artificial Intelligence**: hybrid approaches using NLP and word-embedding techniques such as Word2Vec and BERT. There also strongly appear Knowledge Graphs.
- **Proactive and personalized features**: personify users and deliver recommendations in context.
- They are easy to use, boosting **employee's ability and desire to use digital technology**.
- **Robotic Process Automation (RPA)**.
- Laid on **open-source foundations**.

Search engines require an extensive development to become insight engines. It is obviously hard to meet the market definition.

As stated by the analysts *Stephen Emmott*, *Saniye Alaybeyi* and *Anthony Mullen*, authors of the 2019 Magic Quadrant for Insight Engines, the current competence can be classified into two main categories: **ability to execute** and **completeness of vision**. If we try to connect the dots (each of them representing a product), Google (as a strong challenger) and IBM (as a strong leader), appear quite closed. They achieve the next criteria:

1) A good product.
2) Overall viability.
3) Customer experience.

4) Marketing understanding.
5) Marketing strategy.
6) Innovation.

B. Leaders vs Challengers

Leaders have solid products that offer advanced NLP capabilities, and they enable users to incorporate structured and unstructured data from multiple sources to generate insights. In the other hand, challengers' products are secure, have a broad range of features, and are effective.

Challengers do not have as strong a vision of the insight engine market's future as Leaders do, and their strategy to innovate is not as robust.

In an article posted by IBM, in 2019 [9], they describe its product, **Watson Discovery**, as the IBM's AI-powered search engine, which understands and serves answers from complex business content with context. And what is the success behind it? Well, AI technologies like Natural Language Understanding (NLU), Machine Learning (ML), and Deep Learning (DL) to sort through massive amounts of data and find specific information. It is also continuously tailored through interactions and manual training by a subject matter expert.

Watson Discovery graphical training interface, makes this product even smarter when dealing with company or project specific complex business content, because this interface is easy-to-use and allows the transfer of new specific knowledge to the system [9].

V. Conclusions

Websites content extraction is a hard work that does not just require web scraping. Data cleaning also has to be done during the text normalization (Natural Language Processing), in order to get meaningful content that will be analyzed. Modern search engines do more than this for answering the user questions.

The query terms have different meanings in different contexts, that is why synonyms are a good option for diversifying the scope of the search. And once this wider set of query terms is gotten, what is next is to look for occurrences in the cleaned extracted text, which by using the Pointwise Mutual Information measure, detects if a website is of relevance to the final user or not.

With the presented in this document, the need for talk-ing about linguistics in computing is more than evident. From commercial purposes to more specialized objectives, thesearch engine model here proposed is a good starting point for information analysis.

What can be done as future work, is the programming of an own Knowledge Graph, and the integration of it all in a web application for its distribution. It is a good challenge which reunites another field such as software engineering, and with no doubt it will be a great experience.

References

[1] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*, volume 3rd ed. Pearson, 2007.

[2] Jun Heo, Jaeyeon Won, Yejin Lee, Shivam Bharuka, Jaeyoung Jang, Tae Jun Ham, and Jae W. Lee. Iiu: Specialized architecture for inverted index search. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 1233–1245, New York, NY, USA, 2020. Association for Computing Machinery.

[3] Google. Google search - discover how google search works. https://www.google.com/search/howsearchworks/. (Accessed on 09/15/2020).

[4] Google. How google's algorithm is focused on its users - google search. https://www.google.com/search/howsearchworks/mission/users/. (Accessed on 09/15/2020).

[5] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, Jan 2016.

[6] Common Crawl. In a nutshell, here's who we are. – common crawl. https://commoncrawl.org/about/. (Accessed on 09/26/2020).

[7] Princeton University. Wordnet — a lexical database for english. https://wordnet.princeton.edu/. (Accessed on 09/26/2020).

[8] Gartner. Definition of insight engines - gartner information technology glossary. https://www.gartner.com/en/information-technology/glossary/insight-engines. (Accessed on 09/13/2020).

[9] Luke Palamara. Gartner names ibm a leader in 2019magic quadrant for insight engines - watson blog. https://www.ibm.com/blogs/watson/2019/09/gartner-names-ibm-a-leader-in-2019-magic-quadrant-for-insight-engines/. (Accessed on 09/13/2020).